



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/700,224	11/03/2003	John H. Sandham	1801270.00131US1	3106
23483	7590	12/15/2006	EXAMINER	
WILMER CUTLER PICKERING HALE AND DORR LLP			VO, TED T	
60 STATE STREET			ART UNIT	
BOSTON, MA 02109			PAPER NUMBER	
			2191	

DATE MAILED: 12/15/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

**Office Action Summary**

Application No.

10/700,224

Applicant(s)

SANDHAM ET AL.

Examiner

Ted T. Vo

Art Unit

2191

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☒ Responsive to communication(s) filed on 03 November 2003.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 1-99 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-99 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 03 November 2003 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
- Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
- Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12) ☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☒ None of:
1. ☒ Certified copies of the priority documents have been received.
  2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☒ Information Disclosure Statement(s) (PTO/SB/08)  
Paper No(s)/Mail Date 01/22/04.
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date. \_\_\_\_\_.
- 5) ☐ Notice of Informal Patent Application
- 6) ☐ Other: \_\_\_\_\_.

**DETAILED ACTION**

1. This action is in response to the communication filed on 11/03/2003.  
Claims 1-99 are pending in the application.

***Claim Rejections - 35 USC § 102***

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

3. Claims 1-99 are rejected under 35 U.S.C. 102(b) as being anticipated by Barrio, "Study of the techniques for emulation programming", The Technical University of Catalonia (UPC), Europe, pp. 1-152, 6-2001 (<http://personals.ac.upc.edu/vmoya/docs/emuprog.pdf>).

As per Claim 1: Barrio discloses,

*A method of verifying program code conversion performed by an emulator, comprising the step of: a) executing subject code through an emulator on a subject processor up until a comparable point in the subject code; b) executing the subject code natively on the subject processor up until the same comparable point in the subject code; and c) comparing execution of the subject code natively on the subject processor against execution of the subject code on the subject processor through the emulator at the comparable point in the subject code.*

See page 24, sec. 8: Testing the emulator: (Note: whole reference is also preferred).

Testing is very important in emulation and very hard too. For example to properly test a CPU emulator

Art Unit: 2191

it would mean to generate all the possible instructions (*'executing subject code of the verifying program code'*) which can receive the CPU (claimed step a) and compare the result (claimed step b) in the emulator with the real result (claimed step c). It will be also needed to test combinations of instructions (*'verifying program code conversion'*) because of side effects (flags for example) of the instructions. The interrupt system, the exceptions, the access to memory or to the other devices, if the cycle count is being done correctly, everything should be tested and work well. Something similar happens with the sound and graphic hardware and all the other devices.

As per Claim 20: Barrio discloses claim 20, *A method of verifying program code conversion, comprising the steps of: performing program code conversion to convert subject code into target code through an emulator running on a subject processor and executing the target code to provide an emulated machine state that is stored in a load/store buffer associated with the subject processor; executing the subject code directly on the subject processor to provide a native machine state that is stored in a memory associated with the subject processor; and comparing the emulated machine state contained in the load/store buffer against the native machine state contained in the memory to verify the program code conversion.*

See page 24, sec. 8: Testing the emulator: (Note: whole reference is also preferred)

Testing is very important in emulation and very hard too. For example to properly test a CPU emulator it would mean to generate all the possible instructions (*'performing program code conversion to convert subject code into target code through an emulator running on a subject processor'*) which can receive the CPU and compare the result in the emulator (*comparing the emulated machine state contained in the load/store buffer*)(see Figure 21, p. 41) with the real result (*executing the subject code directly on the subject processor to provide a native machine state*). It will be also needed to test combinations of instructions (*'verifying program code conversion'*) because of side effects (flags for example) of the instructions. The interrupt system, the exceptions, the access to memory or to the other devices, if the cycle count is being done correctly, everything should be tested and work well. Something similar happens with the sound and graphic hardware and all the other devices.

As per Claim 26: Barrio discloses claim 26, *A method of verifying program code conversion (see sec. 8. p. 24), comprising the steps of: first comparing execution of subject code natively on a subject processor against execution of the subject code on the subject processor through a first emulator, thereby verifying program code conversion performed by the first emulator (see in p. 24 as cited above); and once program code conversion performed by the first emulator is verified (The reference CPU emulator is an interpreter see sec. 2. p. 10, therefore its task is translating a program code. This result of a translation is performed and verified as for testing the emulator itself: see P. 24), next comparing execution of subject code*

Art Unit: 2191

*through the first emulator running on the subject processor against execution of the subject code through a second emulator running on a target processor, thereby verifying program code conversion performed by the second emulator using the verified program code conversion performed by the first emulator (See p. 62).*

P. 62: Another interesting use of the free CPU cores is to use them as reference for our own CPU core. The best manner for testing a CPU emulator is to run it hand by hand with the real CPU and look for the differences. This would need some kind of development board with the emulated CPU and many times is impossible to do. A good alternative is to compare the execution of our CPU emulator with another, trusted, CPU emulator.

As per Claim 30: Barrio discloses claim 30, *A method of verifying program code conversion, comprising the steps of: (a) dividing subject code into a plurality of blocks (see p. 19-20, sec 2.2: 'Basic block in compiler theory'), wherein each block includes at least one instruction,*

*(b) executing one the blocks of subject code on a subject processor through a first emulator (see p. 19-20, sec 2.2: 'execute a function or code');*

*(c) comparing execution of the one block of subject code natively on a subject processor against the execution of the one block of subject code on the subject processor through the first emulator, thereby verifying program code conversion of the block of subject code performed by the first emulator (See p. 24, p. 62: discussion of testing the emulator by running the emulator with a program code ('execution of our CPU emulator') and comparing this execution to the execution of a trusted emulator);*

*(d) comparing execution of the same one block of subject code through a second emulator running on a target processor against the already verified execution of the one block of subject code through the first emulator running on the subject processor, thereby verifying program code conversion of the one block of subject code performed by the second emulator (see p. 24, p. 62, "compared"); and*

*(e) repeating steps (b)-(d) for every block of the subject code until program code conversion performed by the second emulator is verified for every block of the subject code (See discussion in p. 64-70, discussion about dynamic translation of an emulator running against basic blocs; see discussion in p. 64-70, discussion about dynamic translation of an emulator running against basic blocks).*

Art Unit: 2191

As per Claim 2: Barrio discloses, *The method of claim 1, wherein: the step (a) comprises executing the subject code on the subject processor up until the comparable point in the subject code through the emulator to provide an emulated machine state (); the step (b) comprises executing the subject code natively on the subject processor up until the same comparable point in the subject code to provide a native machine state; and the step (c) comprises comparing the emulated machine state against the native machine state at every comparable point in the subject code.* See the above-cited passage

As per Claim 3: Barrio discloses, *The method of claim 2, comprising performing the step (a) prior to performing the step (b).* (Barrio's timing shows it can perform either way, i.e. (a) first and (b) second).

As per Claim 4: Barrio discloses, *The method of claim 3, wherein: the step (a) comprises providing an emulated image of the subject processor and/or an emulated image of a memory associated with the subject processor (i.e. state of a CPU emulator, CPU status (p. 26-27); the step (b) comprises providing a native image of the subject processor following the native execution of the program code and/or a native image of the memory associated with the subject processor (i.e. real result run on real CPU emulator (p. 24, or p. 62), following the native execution of the program code; and the step (c) comprises comparing the emulated image of the subject processor against the native image of the subject processor and/or comparing the emulated image of the memory against the native image of the memory. (See p. 24, and p. 62, i.e. 'compared')*

As per Claim 5: Barrio discloses, *The method of claim 4, wherein the step (a) comprises providing the emulated image of the memory in a load/store buffer associated with the memory, such that the memory is not affected by executing the subject code through the emulator (i.e. CPU emulation: emulated image of the memory, see p. 22 first paragraph).*

As per Claim 6: Barrio discloses, *The method of claim 5, wherein the emulated image of the subject processor includes an image of one or more registers (i.e. CPU emulation, p. 22).*

As per Claim 7: Barrio discloses, *The method of claim 6, wherein the emulated image of the subject processor includes an image of one or more condition code flags (e.g. figure 23, p. 48).*

Art Unit: 2191

As per Claim 8: Barrio discloses, *The method of claim 1, comprising executing the subject code natively and through the emulator both within a single process image of the subject processor (see testing an emulator, p. 24, particularly, the emulator is CPU emulator such as VM (p.10)).*

As per Claim 9: Barrio discloses, *The method of claim 8, comprising the step of performing a context switch between at least an emulation context for execution of the subject code through the emulator, and a native context for execution of the subject code natively on the subject processor, the native context and the emulation context being contexts within the single process image (e.g., a CPU (emulator) implements context switching (e.g.. p. 18, sec. 2.1, and p. 43, paragraph 'The getContext()...')).*

As per Claim 10: Barrio discloses, *The method of claim 9, comprising selectively switching between an emulation context for running the emulator on the subject processor, a target execution context for executing target code produced by the emulator on the subject processor, and a subject native context where the subject code runs natively in the subject processor (e.g.. p. 18, sec. 2.1, and p. 43, paragraph 'The getContext()...'))*

As per Claim 11: Barrio discloses, *The method of claim 10, wherein both the native context and the emulation context employ a single image of the subject code (i.e., testing an emulator. Note: when testing an emulator by running against a sample input code, versus a trusted emulator, the running code for two emulators must be the same, i.e., employ a single image of the subject code).*

As per Claim 12: Barrio discloses, *The method of claim 2, comprising: dividing the subject code into a plurality of blocks, wherein each block comprises one of the comparable points in the subject code, executing one of the blocks, and comparing machine states resulting from execution of the one block (refer to "basic block" in the reference, e.g., sec. 2.2).*

As per Claim 13: Barrio discloses, *The method of claim 12, comprising selecting between two or more verification modes, and dividing the subject code into the plurality of blocks according to the selected verification mode (refer to "basic block" in the reference, e.g., sec. 2.2).*

As per Claim 14: Barrio discloses, *The method of claim 13, comprising dividing the subject code into a plurality of blocks, and repeating the executing and comparing steps for each of the plurality of blocks (refer to the rationale addressed in claim 1).*

Art Unit: 2191

As per Claim 15: Barrio discloses, *The method of claim 14, wherein each block comprises any one of: (a) a single instruction of subject code; (b) a basic block comprising a sequence of instructions from a unique entry instruction to a unique exit instruction; or (c) a group block comprising a plurality of the basic blocks (refer to "basic block", sec. 2.2).*

As per Claim 16: Barrio discloses, *The method of claim 1, comprising the steps of: dividing a large segment of the subject code into a plurality of smaller blocks, each block containing one or more instructions from the large segment of subject code; and performing a verification comparison at a block boundary between each pair of consecutive neighbouring blocks in the plurality of blocks (refer to "basic block", sec. 2.2).*

As per Claim 17: Barrio discloses, *The method of claim 16, comprising the steps of: providing the subject processor in an emulation context, where control of the processor rests with the emulator, performing program code conversion on a current block BBn to produce a corresponding block of converted target code, and patching an immediately preceding block of subject code BBn-1 with a return jump; executing a context switch routine to enter a subject native context, and executing the immediately preceding block of subject code BBn-1 natively by the subject processor, such that the executing step terminates with the return jump; executing a context switch routine to return to the emulation context, and performing the verification comparison by comparing a native machine state representing the subject processor following execution of the immediately preceding block BBn-1 with an emulated machine state representing a virtual model of the subject processor held by the emulator following execution of the immediately preceding block BBn-1; executing a context switch to a target execution context, and modelling execution of the target code corresponding to the current block of subject code BBn in the virtual model of the subject processor held by the emulator, thereby leaving the virtual model in a machine state representing the end of the current block BBn; and repeating the above steps for each subsequent block in the plurality of blocks, unless the verification comparison reveals an error in the program code conversion.*

See p. 24-25, "Testing the emulator", where the emulator is a CPU emulator (p. 26) and including CPU emulator core (p. 42, 43, and 44)).



As per Claim 18: Barrio discloses, *The method of claim 17, further comprising restoring the immediately preceding block B<sub>Bn-1</sub> to remove the return jump (only control flow of instruction operations, in a jump table).*

As per Claim 19: Barrio discloses, *The method of claim 1, further comprising the steps of: selecting a block of the subject code; executing the block of subject code on the subject processor through the emulator; and appending a return jump to the block of subject code, and executing the block of subject code natively on the subject processor terminating with the return jump, such that the return jump returns control of the processor to the emulator (e.g., see Fig. 13, p. 34).*

As per Claim 21: Barrio discloses, *selectively inhibiting access by the emulator to the memory associated with the subject processor by buffering load and store requests from the subject processor to the memory in a load/store buffer. See rationale addressed in the rejection of Claim 5.*

As per Claim 22: Barrio discloses, *comprising selectively inhibiting access to the memory when executing the target code, such that an emulated memory image is provided in the load/store buffer. See rationale addressed in the rejection of Claim 5.*

As per Claim 23: Barrio discloses, *The method of claim 20, wherein once the program code conversion performed by the emulator running on the subject processor has been verified, the method further comprising the step of: comparing execution of the subject code through the emulator running on the subject processor against execution of the subject code through a second emulator running on a target processor. See p. 24, and p. 62, comparing the execution of the tested CUP emulator to the trusted CPU emulator).*

As per Claim 24: Barrio discloses, *The method of claim 23, comprising providing a first host processor as the subject processor, and providing a second host processor as the target processor (tested CPU emulator and trusted CPU emulator).*

As per Claim 25: Barrio discloses, *The method of claim 24, wherein the subject code is natively executable on the subject processor whilst not being natively executable on the target processor (See rationale addressed in the rejection of claim 3).*

As per Claim 27: Barrio discloses, *comprising the steps of: performing a first program, code conversion of the subject code including providing a first virtual model of the subject processor in the first emulator, and comparing the first virtual model against the subject processor; and performing a second program code conversion of the subject code including providing a second virtual model of the subject processor in the second emulator, and comparing the first virtual model in the first emulator against the second virtual model in the second emulator* (See rationale addressed in Claim 26, where the CPU emulator is a virtual machine as discussed in p. 10).

As per Claim 28: Barrio discloses, *comprising providing a single way communication from the first emulator to the second emulator.* (See testing the emulator, p. 24, particularly, the emulator is CPU emulator such as VM (p.10)).

As per Claim 29: Barrio discloses, *The method of claim 27, comprising the steps of: synchronizing the first and second virtual models by sending initial state information from the first emulator to the second emulator (i.e. after getting the status from the CPU emulator, its result will be compared the real result); dividing the subject code into a plurality of blocks; for each block of subject code, executing the block of subject code through the first emulator and providing a set of subject machine state data and non-deterministic values to the second emulator; executing the block of subject code in the second emulator substituting the non-deterministic values and providing a set of target machine state data; and comparing the subject machine state data against the target machine state data and reporting an error if a divergence is detected, otherwise repeating the process for a next block of subject code.*

(See p. 19-20, sec 2.2: 'Basic block in compiler theory' and See rationale addressed in the rejection of Claims 2 and 12).

As per Claim 31: Barrio discloses, *The method of claim 30, wherein the subject code is initially divided such that each block of subject code contains a single instruction* (See p. 24, sec. 8, second paragraph, "instructions", or testing some of the parts of the emulation, last paragraph).

Art Unit: 2191

As per Claim 32: Barrio discloses, *The method of claim 31, wherein after program code conversion performed by the second emulator is verified for every block of subject code containing a single instruction, the method further comprises: repeating step (a) by redividing the subject code into a plurality of new blocks, wherein each new block is a basic block comprising a sequence of instructions from a unique entry instruction to a unique exit instruction; and repeating steps (b)-(e) for each basic block, thereby verifying program code conversion performed by the second emulator for every basic block of subject code* (Same rationale as addressed in Claim 30).

As per Claim 33: Barrio discloses, *The method of claim 32, wherein after program code conversion performed by the second emulator is verified for every basic block of subject code, the method further comprises: repeating steps (a) by redividing the subject code into a plurality of group blocks, wherein each group block comprises a plurality of basic blocks; and repeating steps (b)-(e) for each group block, thereby verifying program code conversion performed by the second emulator for every group block of subject code.* (Same rationale as addressed in Claim 30).

As per Claims 34-52: Claims 34-52 claim a computer readable storage medium that stores instruction performing the steps of claims 1-19. See rejection addressed in Claims 1-19.

As per Claims 53-58: Claims 53-58 claim a computer readable storage medium that stores instruction performing the steps of claims 20-25. See rejection addressed in Claims 20-25.

As per Claims 59-62: Claims 59-62 claim a computer readable storage medium that stores instruction performing the steps of claims 26-29. See rejection addressed in Claims 26-29.

As per Claims 63-66: Claims 63-66 claim a computer readable storage medium that stores instruction performing the steps of claims 30-33. See rejection addressed in Claims 30-33.

As per Claims 67-85: Claims 67-85 claim an emulator apparatus that performs the steps of claims 1-19. See rejection addressed in Claims 1-19.

As per Claims 86-91: Claims 86-91 claim an emulator apparatus that performs the steps of claims 20-25. See rejection addressed in Claims 20-25.

Art Unit: 2191

As per Claims 92-95: Claims 92-95 claim an emulator apparatus that performs the steps of claims 26-29.

See rejection addressed in Claims 26-29.

As per Claims 96-99: Claims 96-99 claim an emulator apparatus that performs the steps of claims 30-33.

See rejection addressed in Claims 30-33.

**Conclusion**


4. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ted T. Vo whose telephone number is (571) 272-3706. The examiner can normally be reached on 8:00AM to 4:30PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Y. Zhen can be reached on (571) 272-3708.

The facsimile number for the organization where this application or proceeding is assigned is the Central Facsimile number **571-273-8300**.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

TTV  
December 08, 2006

  
**TED VO**  
**PRIMARY EXAMINER**  
**TECHNOLOGY CENTER 2100**